

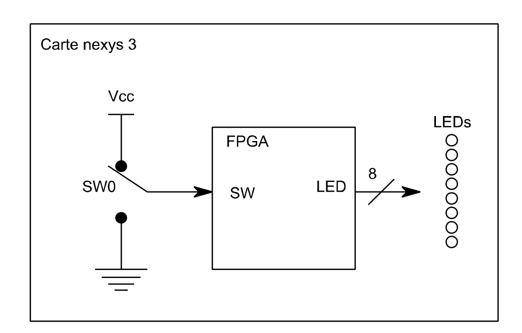
Filière ELT – 2ème année

TP1
La prise en main de la carte nexys 3

Au cours de ce TP nous allons apprendre comment un programme VHDL s'écrit dans l'environnement ISE, et ensuite, comment ce programme se synthétise et s'implante dans une cible donnée. Ici, nous utiliserons la carte « Nexys 3 » de Xilinx où le FPGA en question est le Spartan 6: XC6SLX16 (324 broches).

Le manuel d'utilisation de cette carte se trouve sur le bureau. Vous pouvez aussi le télécharger à partir du site fabricant www.digilentinc.com.

Travail 1 : Allumer un motif sur les 8 LED de la carte (voir la vidéo se trouvant sur la page web de TP)



Dans l'environnement ISE, faire une entité puis l'architecture correspondante pour allumer le modèle suivant « 11001010 » sur les 8 LED de la carte si l'interrupteur SW0='1' sinon « 11110000 ».

Ajoutez à ce projet des contraintes liées aux broches physiques de FPGA. Pour cela, reportez-vous à la page 18 du manuel d'utilisation de la carte.

Utilisant l'outil ADEPT, chargez le FPGA et vérifiez si le circuit fonctionne comme vous le souhaitiez.

Travail 2 : allumage sur les 7-segments

Dans un premier temps, nous allons afficher sur un des 7-segments, un chiffre hexa décimal entre 0 et F. Le circuit à concevoir est le suivant :

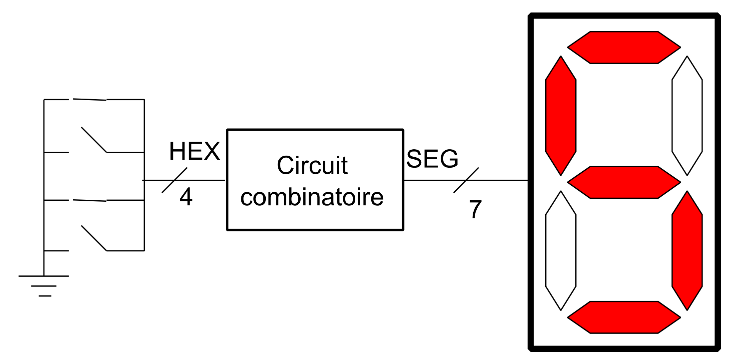


Figure 1- Le circuit à réaliser

Les connexions de la carte nexys 3 sont présentées ci-dessous :

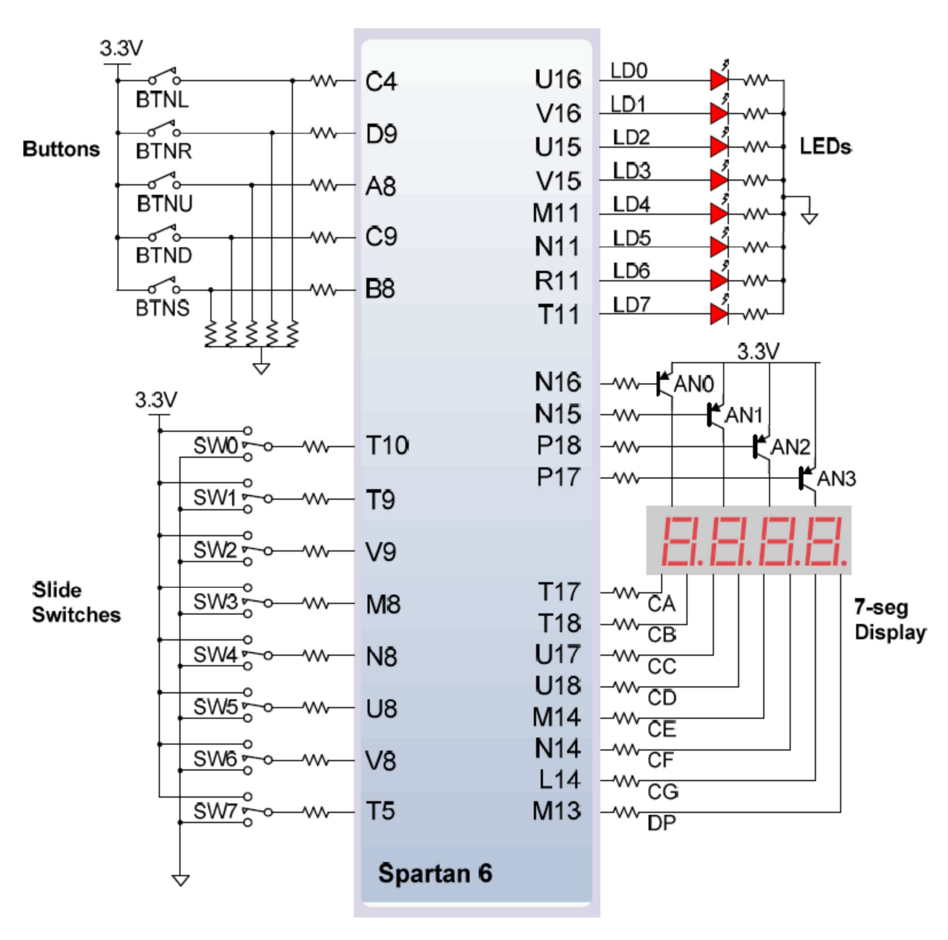
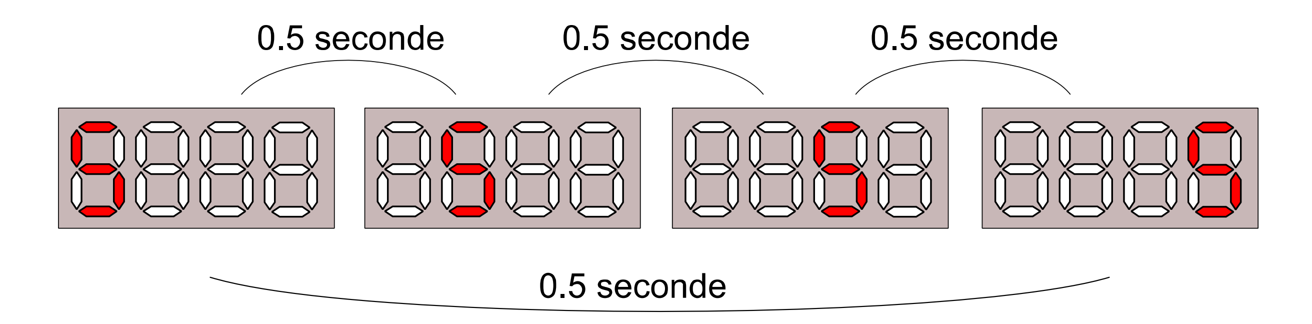


Figure 2- Schématique de la carte nexys3

Pour les 4 interrupteurs de la Figure 1 de on utilise les SW0 – SW3 du schéma de la Figure 2. Le circuit combinatoire, présenté sur la Figure 1, peut se réaliser facilement par l'instruction with-select (ou when-else). Pour pouvoir allumer un 7–segments particulier parmi les 4, il faudra qu'on met à 0 volt la ligne AN correspondant. Par exemple pour allumer une valeur uniquement sur le 7-seg de gauche, on envoie un '0' logique sur AN0, et un '1' logique AN1, AN2 et AN3. Si on souhaite afficher la valeur sur tous les 4 7-segment, on envoie un '0' logique sur les 4 lignes AN0-AN3.
On peut définir AN comme un std_logic_vector(3 downto 0) et saisir : AN <= "0000". Si vous voulez allumer uniquement le segment de gauche: AN <= "0111".

Travail 3 :
Faite défiler le nombre hexa sur les 4 segments comme il est présenté sur la figure ci-dessous. Le temps d'affichage par sept-segment est de 0,5 sec.



Comment faire ?
Une idée est d'utiliser le même circuit qu'avant mais on essaye de ne pas envoyer la même valeur sur les lignes AN. Par exemple j'envoie sur AN la valeur "0111" et je temporise 0.5 seconde, puis j'envoie "1011" et je temporise 0.5 seconde et ainsi de suite. Il faudra donc créer une base temporelle qui me donne toutes les 0.5 seconde un signal de déclanchement qu'on appelle "*triger*". Sur ce *triger* je change de valeur que je dois envoyer sur AN. Ce principe est présenté par un chronogramme. Les chronogrammes constituent un moyen très répandu pour définir les actions qui se déroule dans le temps.

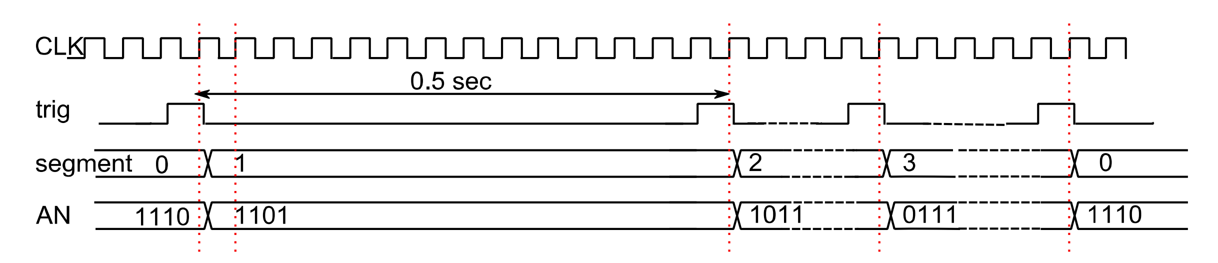


Figure 3- Chronogramme conçu

Une fois le chronogramme est conçu, on imagine un circuit contenant des blocs qui réalisent des fonctions souhaitées. Il est **très important** de dessiner le diagramme fonctionnel (ou architecture) détaillé de votre circuit avant de commencer à programmer. La Figure 4

présente l'architecture du circuit. On génère un signal "trig" avec une période de 0.5 second à partir de l'horloge de la carte qui est à 100 MHz. Ce signal "trig" va servir pour commuter entre les 7-segments. Pour simplifier, on va les numéroter de 0 à 3 avec un signal de type entier (signal "segment sur la figure) qui garde la trace de 7-segment allumé. Ensuite, un circuit combinatoire qui utilise ce signal "segment" et qui envoie la bonne valeur sur AN.

segment	AN
0	"0111"
1	"1011"
2	"1101"
3	"1110"

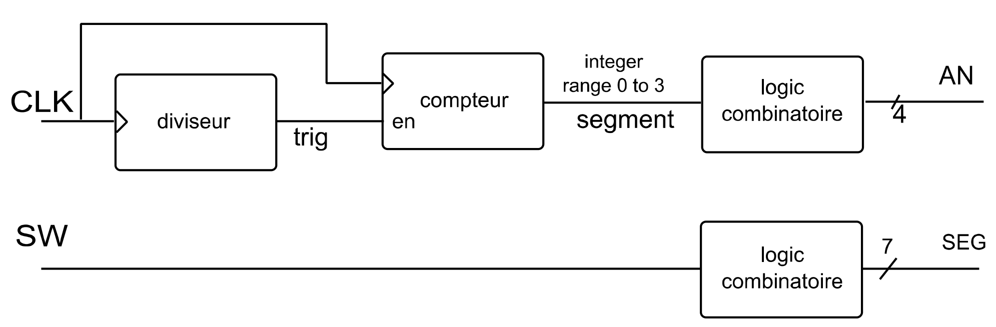


Figure 4- diagramme fonctionnel (architecture) du circuit

Réalisation du diviseur :

La fréquence d'horloge étant à 100 MHz (période est de 10 ns), pour obtenir 0.5 s, il faudra 0.5s/10ns=50.000.000 périodes 10 ns.

```
Signal comp : integer range 0 to 50000000 ; -- comp est un
                  --signal qui sera représenté sur 26 bits.
                  --Il peut aller donc jusqu'à 2^26-1=67.108.863
…
diviseur : process(CLK)
begin
      if CLK'event and CLK='1' then
            comp <= comp + 1;
            if comp = 49_999_999 then
                  trig <= '1';
                  comp <= 0;
            else
                  trig <= '0';
            end if;
      end if;
end process;
```

La réalisation du compteur :

C’est un circuit synchronisé avec CLK mais activé avec trig :

```
Signal segment : integer range 0 to 3 ; -- segment donc sera
                  --représenté par 2 bits, il ne peut aller que
                  -- jusqu’à 3. Ainsi 3+1 fera 0 et non pas 4.
compteur:process (CLK)
begin
      if CLK’event and CLK=’1’ then
            if trig = ‘1’ then
                  segment <= segment + 1;
            end if;
      end if;
end process;
```

logique combinatoire :

Se réalise avec une commande with-select ou when-else.

Suite du travail

Changez la période de 0,5 seconde par 0,5 milliseconde et observez le résultat. En effet, on voit toutes les LED allumées grâce à la persistance rétinienne. Cette manière d’affichage s’appelle « l’affichage multiplexé » et a comme but de réduire le nombre de fils nécessaires pour allumer toutes les combinaisons : ici 11 fils au lieu de 28 (pourquoi).

Travail 4 : Compteur 4 digits.

Modifiez le circuit précédent pour générer en interne le signal HEX et afficher finalement un compteur hexadécimal 4 digits, qui est incrémenté toutes les 0,1 seconde.

0000→0001→0002→… →000F→0010→0011→…→FFFF→0000→0001→…

Oh là là, comment faire ??!!

Il faudra toujours pouvoir repérer le segment allumé, donc la partie qui générait le signal "trig" et "segment" ne sera pas touchée. En termes d'affichage, on affichait la valeur fixée par les interrupteurs alors que maintenant il faudra la générer.

On pense à une registre de taille 4 digits hexadécimaux (c’est-à-dire 16 bits) que l'on incrémente toutes les 0.1 seconde. Ce registre peut déclarer comme

```
signal compt : std_logic_vector (15 downto 0);
```

Il nous faut un autre signal "triger" qui *trigue* toutes les 0.1 second. Maintenant il nous faut juste un circuit combinatoire qui envoie les différents digits du registre 16 bits au bon moment vers les 7-segment. Voir le tableau ci-dessous:

segment	AN	SEG

0	0111	compt(15 downto 12)
1	1011	compt(11 downto 8)
2	1101	compt(7 downto 4)
3	1110	compt(3 downto 0)

La Figure 5 présente l'architecture détaillée du circuit.

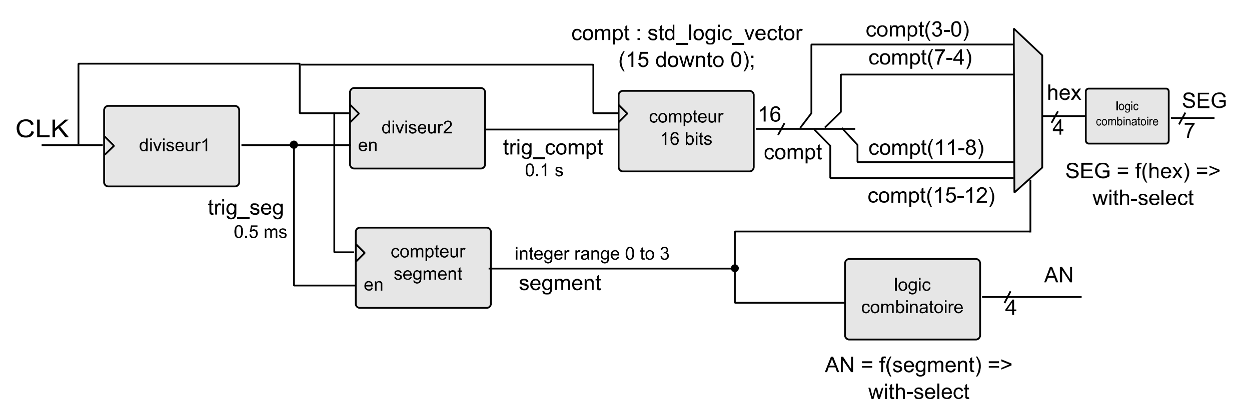


Figure 5- Architecture du circuit

Remarques :

- Dans le "diviseur1", le compteur est un entier allant de 0 à (0.5ms/10ns)=50000
  - o `constant trig_const : integer := 50000;`
  - o `signal comp : integer range 0 to trig_const:=0;`
- Dans le circuit compteur, le comptage de "segment" se fait tous les 200 cycles de "trig".
- Segment est un entier de 0 à 3
  - o `signal segment : integer range 0 to 3:=0;`
- compt est un std_logic_vector(15 downto 0)

Travail 5 : Simulation

Nous allons effectuer une simulation pour l’exemple précédent et visualiser les différents signaux de design. Afin de ne pas trop ralentir la simulation, on remplace le temps 0.5 ms par 40 ns, et 0.5 seconde par 8 microsecondes (on garde toujours le rapport 200).

Faire un « new source » de type « VHDL testbench ». Ce testbench est un fichier de test pour le module du travail 4. Dans ce module, il suffira juste de créer une horloge de fréquence 100 MHz :

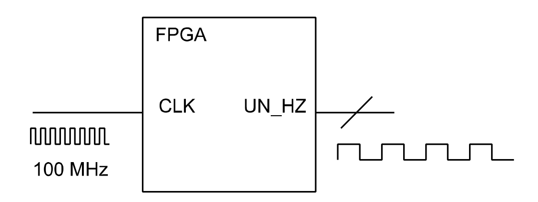
```
BEGIN
   uut: compteur PORT MAP (
          MCLK => MCLK,
          AN => AN,
          SEG => SEG
         );
```

```
    MCLK_process :process
    begin
            MCLK <= '0';
            wait for 5 ns;
            MCLK <= '1';
            wait for 5 ns;
    end process;
END;
```

Dans la fenêtre « Design » et sous la rubrique « view », changer « implementation» en « Simulation ». Effectuer une simulation comportementale jusqu'à 10 microsecondes. Faites afficher les différents signaux de votre circuit.

Exercices : Petits travaux d'entrainement

1- Génération d'un signal carré d'un Hz :



Nous souhaitons diviser la fréquence de l'horloge 100 MHz de la carte pour générer un signal carré 1 Hz. Pour cela il faudra d'abord créer un compteur qui s'incrémente à la vitesse de l'horloge de la carte et dès qu'il atteint une valeur pre-déterminée, la sortie bascule (change d'état).

L'algorithme ci-dessous peut être utile :

```
Sur le front montant de l'horloge 100 MHz
        Incrémenter un compteur
        Si le compteur atteint la valeur représentant 0,5 seconde
                Initialiser le compteur à zéro
                Basculer la sortie (UN_HZ  <= not  UN_HZ)
        end si
end
```

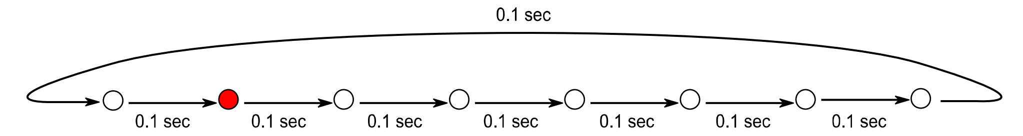
Remarque : On définit le compteur en tant qu'un entier :

```
signal compt : integer range 0 to 50000000 ;
```

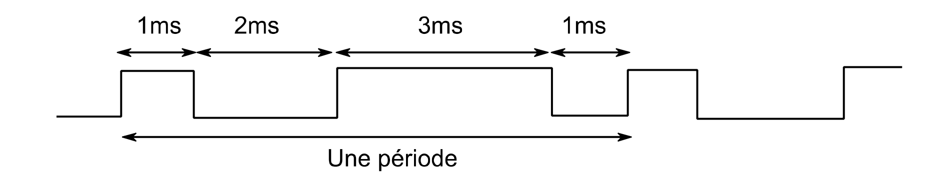
Pour visualiser ce signal, connecter toutes les 8 LED au signal UN_HZ. Utiliser par exemple la commande concurrente ci-dessous :

```
LED <= "1111_1111" when UN_HZ = '1' else "0000_0000";
```

2- Faire un chenillard sur les LED



3- Créez le signal ci-dessous et faite le sortir les sur la broche 1 du connecteur JD1.



4- Le but est de créer un jeu. Faites afficher « 8888 » sur les segments. Dès que les segments s'éteignent, deux joueurs appuient sur leurs boutons correspondant. Celui qui réussit de le faire en premier voit sa LED allumée et elle reste allumée jusqu'à ce que l'animateur appuie sur le bouton RESET.

5- Faire un circuit pour simuler deux dés. A chaque appuie sur le bouton BTND, sur deux des segments sont affichés pendant 1 seconde 00 et ensuite deux nombres aléatoires entre 1 et 6.